NAME

vxlan - Virtual eXtensible LAN interface

SYNOPSIS

To compile this driver into the kernel, place the following line in your kernel configuration file:

```
device vxlan
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
if_vxlan_load="YES"
```

DESCRIPTION

The **vxlan** driver creates a virtual tunnel endpoint in a **vxlan** segment. A **vxlan** segment is a virtual Layer 2 (Ethernet) network that is overlaid in a Layer 3 (IP/UDP) network. **vxlan** is analogous to vlan(4) but is designed to be better suited for large, multiple tenant data center environments.

Each **vxlan** interface is created at runtime using interface cloning. This is most easily done with the ifconfig(8) **create** command or using the *cloned_interfaces* variable in rc.conf(5). The interface may be removed with the ifconfig(8) **destroy** command.

The **vxlan** driver creates a pseudo Ethernet network interface that supports the usual network ioctl(2)s and thus can be used with ifconfig(8) like any other Ethernet interface. The **vxlan** interface encapsulates the Ethernet frame by prepending IP/UDP and **vxlan** headers. Thus, the encapsulated (inner) frame is able to be transmitted over a routed, Layer 3 network to the remote host.

The **vxlan** interface may be configured in either unicast or multicast mode. When in unicast mode, the interface creates a tunnel to a single remote host, and all traffic is transmitted to that host. When in multicast mode, the interface joins an IP multicast group, and receives packets sent to the group address, and transmits packets to either the multicast group address, or directly to the remote host if there is an appropriate forwarding table entry.

When the **vxlan** interface is brought up, a udp(4) socket(9) is created based on the configuration, such as the local address for unicast mode or the group address for multicast mode, and the listening (local) port number. Since multiple **vxlan** interfaces may be created that either use the same local address or join the same group address, and use the same port, the driver may share a socket among multiple interfaces. However, each interface within a socket must belong to a unique **vxlan** segment. The analogous vlan(4) configuration would be a physical interface configured as the parent device for multiple VLAN interfaces, each with a unique VLAN tag. Each **vxlan** segment is identified by a 24-bit value in the **vxlan** header called the "VXLAN Network Identifier", or VNI.

When configured with the ifconfig(8) **vxlanlearn** parameter, the interface dynamically creates forwarding table entries from received packets. An entry in the forwarding table maps the inner source MAC address to the outer remote IP address. During transmit, the interface attempts to lookup an entry for the encapsulated destination MAC address. If an entry is found, the IP address in the entry is used to directly transmit the encapsulated frame to the destination. Otherwise, when configured in multicast mode, the interface must flood the frame to all hosts in the group. The maximum number of entries in the table is configurable with the ifconfig(8) **vxlanmaxaddr** command. Stale entries in the table are periodically pruned. The timeout is configurable with the ifconfig(8) **vxlantimeout** command. The table may be viewed with the sysctl(8) **net.link.vxlan.N.ftable.dump** command.

MTU

Since the **vxlan** interface encapsulates the Ethernet frame with an IP, UDP, and **vxlan** header, the resulting frame may be larger than the MTU of the physical network. The **vxlan** specification recommends the physical network MTU be configured to use jumbo frames to accommodate the encapsulated frame size.

By default, the **vxlan** driver sets its MTU to usual ethernet MTU of 1500 bytes, reduced by the size of vxlan headers prepended to the encapsulated packets.

Alternatively, the ifconfig(8) **mtu** command may be used to set the fixed MTU size on the **vxlan** interface to allow the encapsulated frame to fit in the current MTU of the physical network. If the **mtu** command was used, system no longer adjust the **vxlan** interface MTU on routing or address changes.

HARDWARE

The **vxlan** driver supports hardware checksum offload (receive and transmit) and TSO on the encapsulated traffic over physical interfaces that support these features. The **vxlan** interface examines the **vxlandev** interface, if one is specified, or the interface hosting the **vxlanlocal** address, and configures its capabilities based on the hardware offload capabilities of that physical interface. If multiple physical interfaces will transmit or receive traffic for the **vxlan** then they all must have the same hardware capabilities. The transmit routine of a **vxlan** interface may fail with ENXIO if an outbound physical interface does not support an offload that the **vxlan** interface is requesting. This can happen if there are multiple physical interfaces involved, with different hardware capabilities, or an interface capability was disabled after the **vxlan** interface had already started.

At present, these devices are capable of generating checksums and performing TSO on the inner frames in hardware: cxgbe(4).

EXAMPLES

Create a **vxlan** interface in unicast mode with the **vxlanlocal** tunnel address of 192.168.100.1, and the **vxlanremote** tunnel address of 192.168.100.2.

```
    ifconfig vxlan create vxlanid 108 vxlanlocal 192.168.100.1 vxlanremote 192.168.100.2
```

Create a **vxlan** interface in multicast mode, with the **local** address of 192.168.10.95, and the **group** address of 224.0.2.6. The em0 interface will be used to transmit multicast packets.

```
    ifconfig vxlan create vxlanid 42 vxlanlocal 192.168.10.95 vxlangroup 224.0.2.6 vxlandev em0
```

Once created, the **vxlan** interface can be configured with ifconfig(8).

The following when placed in the file */etc/rc.conf* will cause a vxlan interface called "vxlan0" to be created, and will configure the interface in unicast mode.

```
    cloned_interfaces="vxlan0"
    create_args_vxlan0="vxlanid 108 vxlanlocal 192.168.100.1 vxlanremote 192.168.100.2"
```

SEE ALSO

inet(4), inet6(4), vlan(4), rc.conf(5), ifconfig(8), sysctl(8)

M. Mahalingam and et al, *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*, August 2014, RFC 7348.

AUTHORS

The **vxlan** driver was written by Bryan Venteicher <bryanv@freebsd.org>. Support for stateless hardware offloads was added by Navdeep Parhar <np@freebsd.org> in FreeBSD 13.0.